

## **Great Expectations :: Prediction in Entertainment Applications**

Robert Burke

MediaLabEurope  
rob@mle.ie

The entertainment world is teeming with applications for expressive, adaptive agents. Many such applications feature some sort of “creatures” – anthropomorphic or non – that must maintain the illusion of life while interacting with one another, as well as with human participants. This chapter discusses approaches to implementing these creatures. It begins by describing an agent-based architecture representative of many of those found in the entertainment world, and then shows how the integration of a new representation for prediction allows new forms of learning, adaptation and expressive behavior for agents that use that architecture.

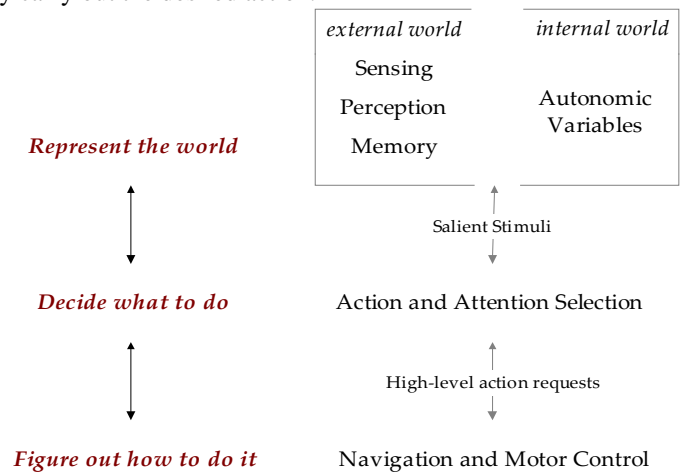
### **The Usual Suspects**

Here are some examples of agents in entertainment applications:

- An anthropomorphic robot uses cameras and microphones to perceive real-world visual and audio information. It detects a person standing in front of it, and tries to mimic that person’s upper body motion as best it can. It uses facial expressions to show its satisfaction, confusion, curiosity, and so on. Sometimes, when “feeling mischievous,” it instead performs other gestures it has remembered from previous interactions.
- In an interactive installation, an autonomous virtual sheepdog exists on a field with other virtual creatures and objects. Human participants act as the shepherd, shouting voice commands which the virtual dog must interpret to successfully herd sheep.
- A fearsome behemoth named Goatzilla is a virtual creature that inhabits the Scottish highlands. He begins his life with no knowledge of how to scavenge for food. In order to survive, he must learn – in real time, while maintaining the illusion of life – strategies for rustling up the local shepherd’s tasty sheep.
- In a virtual sports simulator, an autonomous agent controls the right-wing forward player on a hockey team. This agent perceives the state of the arena from its vantage point on the ice, and, given its knowledge of the game and the social dynamics of the team, including offensive strategies practiced with the other autonomous players, determines an appropriate course of action.

## I Perceive, I Decide, I Do

These four agents, while different in purpose, have quite a lot in common. Each is expressive. Each should be robust to changes in its dynamic world. And, practically, on a moment-to-moment basis, each must work in real-time to choose between a variety of possible actions. To do so, each agent perceives its world (real and/or virtual), and bases its decisions on both its current perception of the context, as well as its internal state. By internal state, we mean for example that the sheepdog might be more enthusiastic if very hungry, and the hockey player might adopt a defensive posture if recovering from an injury. Once each agent decides what to do, it must work with the degrees of freedom available to it to expressively carry out the desired action.



**Fig. 1.** High-level view of a layered cognitive architecture for an autonomous agent. See [6] or [17] for more detail.

The details of the last task – the agent’s act of actually modifying the world– are largely task-specific and vary greatly between applications. An animated character, for example, must figure out how to move expressively and remain in character while performing a particular action ([27], [26], [11]). A robot might react similarly, although with servos in the real world ([4], [32]). Agents are often constrained by the laws of either real or cartoon physics: the virtual hockey player, for example, can’t stop skating on a dime.

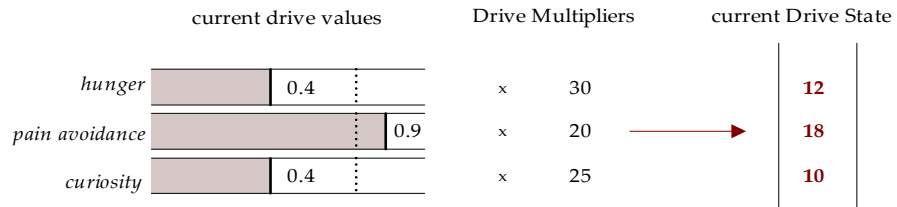
## What Motivates You?

Just as, in stage drama, a method actor might ask “what is my motivation?” an agent can be aware of what motivates it to behave a particular way.

One obvious way in which the simplest goals in an agent can be represented is in terms of *drives*. An agent might have a number of drives, such as hunger,

pain avoidance, reducing the distance to the opponent’s goal, and so on. Each drive could potentially be reduced to a simple scalar number we’ll call the Drive Output, the value of which can be obtained by some function of other aspects of the agent’s system. In a virtual dog, for instance, hunger might simply be a function of the amount of food in the creature’s virtual stomach. Or, it might also take into account things like whether or not the creature can perceive food, what time of day it is, and so on. The output of each drive can then be arbitrarily scaled to be between 0 and 1. At 0, the drive is considered fully satisfied [30].

It’s then easy to preferentially weight one of the drives over another, by including a Drive Multiplier – a scalar unique to each drive that is multiplied by the Drive Output. These Multipliers can change over time, causing the creature to occasionally favor one drive or another. They could, for example, reflect a circadian (daily) rhythm that causes an agent to weight a “sleep” drive more in the evening.



**Fig. 2.** Three Drives, their Drive Multipliers, and the resulting Drive State.

A greater challenge is representing more complex goals. For instance, the desire to explore new alternatives might constitute a derived “Curiosity Drive.” If, for some reason, it seems like curiosity should be represented on a different, more abstract level than hunger, perhaps it’s because it’s harder to pinpoint the root physiological cause of human curiosity. In entertainment applications, it can make practical sense to represent abstract goals as drives. The hockey player, in the context of the social activity of playing hockey, might have a goal to prevent the opposing team from taking a shot on net, which is manifested as a Drive that is particularly intense when the goalie is exhausted.

### Perceive the World to find ways to satisfy drives

So our agent now has a motivation: it wants to reduce its Drives. In order to do so, it will need to perceive the current state of the world to find relevant information. If the drive state represents a sort of *internal context* for the agent, the perceived world state is the corresponding *external context*.

For many applications, it is useful to distinguish between sensing and perception. The distinction is obvious in the physical world – the robot’s cameras and microphones sense, and the interpretation of the signals sensed by those devices constitutes perception. The hockey player may need to base his decisions on incomplete visual information, instead of having full knowledge of the status of all other players on the ice. This facilitates surprise, and motivates the inclusion of a spatial working memory, such as the one described in ([17]).

Perception is a rich area of research. Indeed, Rodney Brooks sparked a revolution of sorts in artificial intelligence by suggesting that intelligence could be achieved primarily through perception, rather than representation ([5]). Although few AI practitioners today would agree that representation is *un*-necessary, there is plenty of innovation that demonstrates how intelligence can arise from thoughtful perceptual representations. Isla, for instance, has designed perceptual representations that allow a creature to reason about object persistence, by combining a spatial ‘working memory’ with an ability to predict how the location of unseen objects in the world will change with time. ([16], [21])

One useful way to classify perceptual input is by using a hierarchical tree of “Percepts.” Every nugget of sensory information passes through the Percept Tree and activates specific Percepts. The activations of the Percepts can be used as input to the creature’s action selection mechanism.

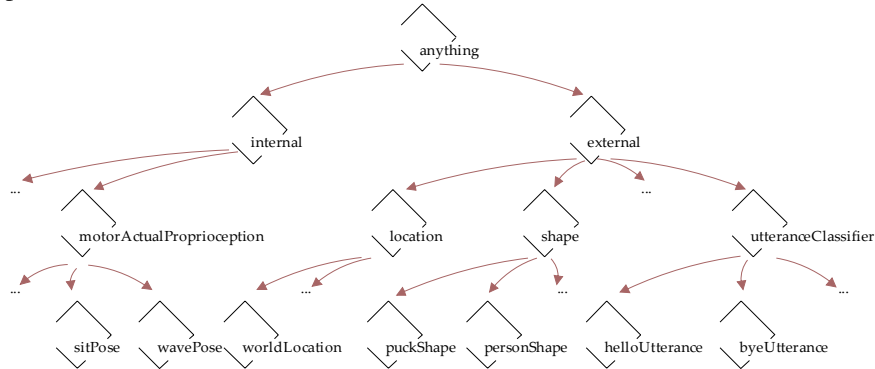
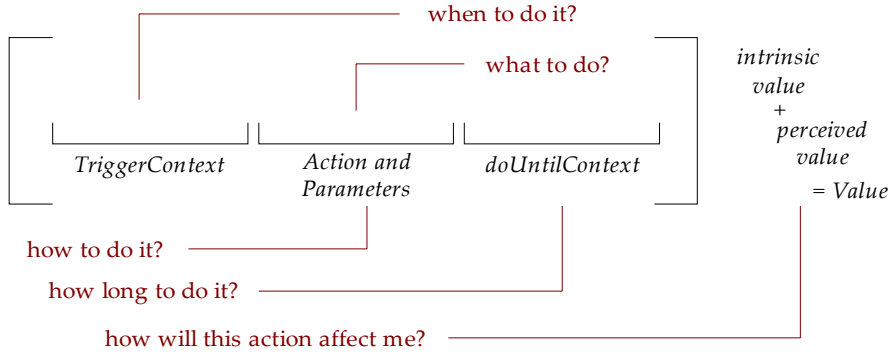


Fig. 3. Sample of part of an agent’s Percept Tree.

### Deciding What to Do Can Be Reactive

One way the agent can make decisions is by simply *reacting* to the context it perceives. In other words, the creature perceives a particular context (both internal and external), weighs a variety of options based on their perceived values, and then makes an informed probabilistic decision. In order to make this kind of decision, the creature needs a representation that will allow it to answer the question, “in a given context, if I perform a particular action, how good will that be for me?”

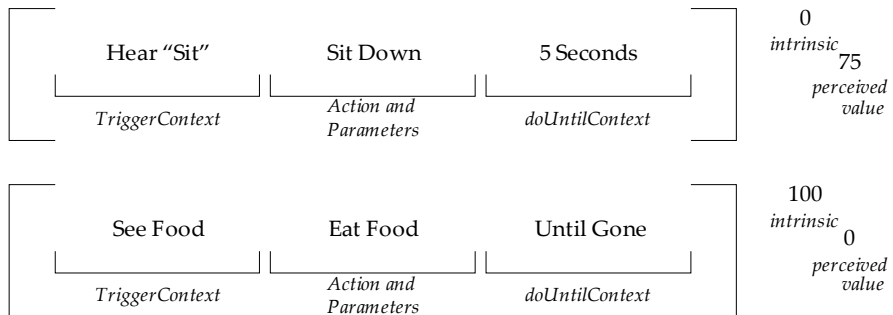
Blumberg accomplishes this with a representation called the *ActionTuple*. [17] It is a useful starting point for a discussion of action selection, as it encapsulates and formalizes components found in many such mechanisms in the entertainment realm (and beyond). An ActionTuple consists of four components:



**Fig. 4.** Anatomy of Blumberg’s ActionTuple.

The *TriggerContext* is the world state in which the ActionTuple is valid. The *Action* is performed by the agent if this ActionTuple is activated, and its *Parameters* describe the details of that action. A typical parameter is the object on which the action is performed: “pick up *the ball*” or “sit *facing the shepherd*.” The *DoUntil* context describes for how long the ActionTuple should remain active. And the *Value* represents, in relation to the Drives, how valuable it is to perform this action. The *Value* can be either Intrinsic (and fixed) or Perceived (and potentially flexible). (For more on mapping Drives to Value, see [23].)

Blumberg uses a dog-training paradigm to demonstrate how the ActionTuple representation can support a virtual creature that learns to perform tricks to receive rewards [3]. Suppose we want to construct a virtual dog that will sit when it hears the word “sit.” When the dog does so, we will reward it with a food treat that it should promptly eat in order to satisfy its hunger drive. We can produce this behavior using two ActionTuples: one that says “when I hear the word ‘sit,’ and I sit down for some time, it’s pretty good for me,” and another that says “when I see food, and I eat it, it’s *really* good for me.” Both of these are depicted below.



**Fig. 5.** Two ActionTuples which, taken together, suggest that sitting when you hear the word “sit” has a moderately high perceived value (top), and eating food when you see it has an even higher intrinsic value (bottom).

On a moment-to-moment basis, the dog must react to its perceptions by exclusively selecting one of a number of possible ActionTuples, including these two. Most of the others will have low Values. So if the dog hears the word “sit,” then

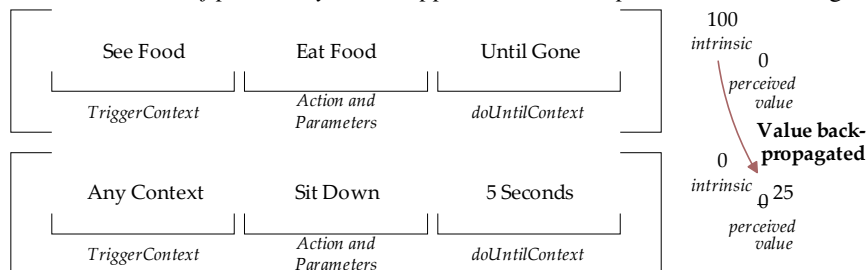
with high probability, it will select the first ActionTuple depicted above, and consequently sit down.

## Adaptations

For many entertainment applications, agents with completely pre-defined behavior rules are appropriate. However, we've recently seen increasingly complex and interesting agents that are able to *adapt* to changes in their worlds. There are plenty of reasons this is exciting. A virtual character can be *personalizable*, so that it gets to know *you* as you get to know it. An agent that's meant to act as a companion or friend can thus learn how to be helpful. A combatant, on the other hand, can *innovate* new a personalized counter-attack.

Blumberg was inspired by an approach used to train real animals to integrate a form of adaptation into the ActionTuple representation. The technique requires two degrees of freedom: first, the *Value* of some ActionTuples must be plastic, so that Tuples that previously were not considered valuable could come to be seen as useful (and vice versa). Second, we must be able to replicate ActionTuples that have *modified Trigger Contexts*, so that the agent can learn the context in which an action is valuable.

The agent begins its life with several “consummatory” ActionTuples that have fixed *Intrinsic Values*, such as “Eat food.” It also begins with other “appetitive” ActionTuples with flexible *Perceived Values*. The Intrinsic Value is *back-propagated* from the consummatory ActionTuples when they are active, *into the Perceived Value of previously-active appetitive ActionTuples*, as shown in Fig. 6.

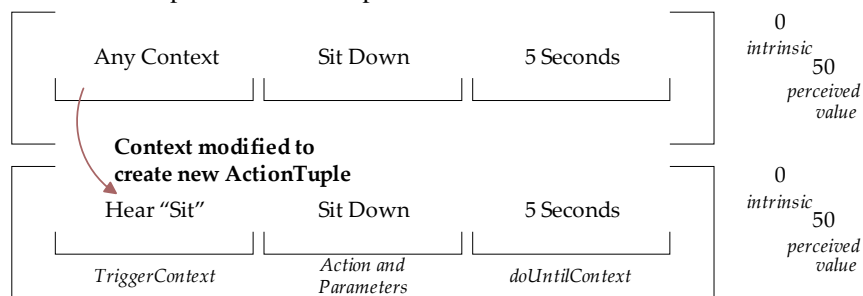


**Fig. 6.** Back-propagation of value. Part of the “Eat food” ActionTuple’s total Value, 100, is back-propagated to the “Sit in any context” ActionTuple’s Perceived Value.

In this manner, an agent can start with a simple ActionTuple like the one above, that makes it sit down in any context. In the absence of food, this dog-agent is unable to activate any high-valued consummatory ActionTuples, and so it is left to “explore” the low-valued ActionTuples like the one above. At some point, the dog randomly sits down, and, lo and behold, food appears, making it possible to activate the “eat the food” ActionTuple. Some of the value from the “eat the food” ActionTuple is *back-propagated* to the “In any context, sit down” ActionTuple, so that its “Perceived Value” increases (as is occurring in Fig. 6.). Of course, its *Intrinsic Value* is (and will always be) 0, since the act of sitting down itself doesn’t assist the dog in any way. But the back propagation, as depicted in

the figure, allows the act of sitting down to gain *perceived* value because it leads with some reliability to a reward.

When an appetitive ActionTuple achieves a high Perceived value, it can refine the TriggerContext to construct ActionTuples that will achieve higher perceived values, and thus be more useful. It does this by determining which Percepts were reliable indicators of successful trials. In this dog training case, if the dog has heard an utterance that sounds like “sit” recently, then sitting is likely to lead to a reward. Thus the presence of the “hearing the sit sound” stimulus is a reliable indicator for the success of the trial, and by innovating a *new ActionTuple* that reflects this, the dog will be able to represent the fact that the sitting action is *more* valuable when performed in that particular context.



**Fig. 7.** The TriggerContext of the ActionTuple is modified to create a new, more specific (and soon-to-be higher-valued) ActionTuple – one that is only triggered when you hear an utterance that sounds like “sit.”

### The Importance of Expectation

Using the back-propagation technique just described, behaviors that were previously perceived as neutral can come to acquire value when they are performed in particular contexts. Although this produces the desired effect of causing the agent to favor reliably appetitive behaviors, it leaves the agent incapable of answering “why do I expect this appetitive behavior will be valuable?”

In other words, there’s no capacity for *expectation*. When a living dog sits down after hearing the word sit, it gives you an intense stare laden with expectation: “Where’s my treat, buddy?” If the treat comes, the dog is satisfied and his expectations are confirmed. If it doesn’t, there’s an *expectation violation* and the dog is left wondering what went wrong. Then, a few moments later, if you give the dog some food for no apparent reason, he is *surprised* and possibly wonders if there’s something he could do in future to produce the same result.

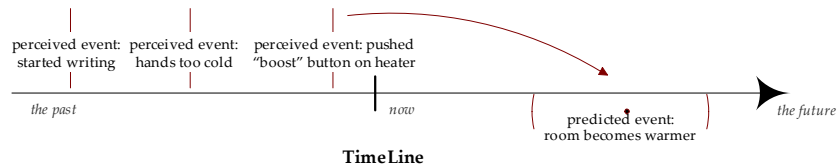
An agent’s ability to expect affords rich opportunities for learning. Imagine we had a system that was capable of predicting the occurrence of salient events in the world. It would be able to *expect* how the world is going to change, and modify its behavior accordingly. It would even be able to predict how *its own actions* will change the world before it performs them. And, since the world will inevitably prove the agent’s expectations to be wrong on occasion, the agent would be able

to use unexpected events – both surprises and expectation violations – to motivate learning.

Not only can the capacity for expectation motivate an agent to learn, but it also provides rich opportunities for producing the illusion of life. An agent can emote its eager *anticipation* of something it expects will make its world a better place, and, if necessary, even take steps to facilitate the event's occurrence. Conversely, if it expects that something about to happen will make the world worse, it can dread the event, try to escape, and emote appropriately. When things don't go as predicted, the resulting expectation violations might cause *frustration* or *relief*. And *surprises*, depending on their effect on drive state, might be perceived as pleasant or unpleasant.

### But how to predict things?

In order to form expectations, an agent needs some understanding of causality. Or, more accurately, it needs some representation of *apparent* causality. An important and often implicit component of every expectation is a sense of timing. It will be helpful to think of our agent as being able to represent events, both perceived and predicted, on some sort of *TimeLine*:



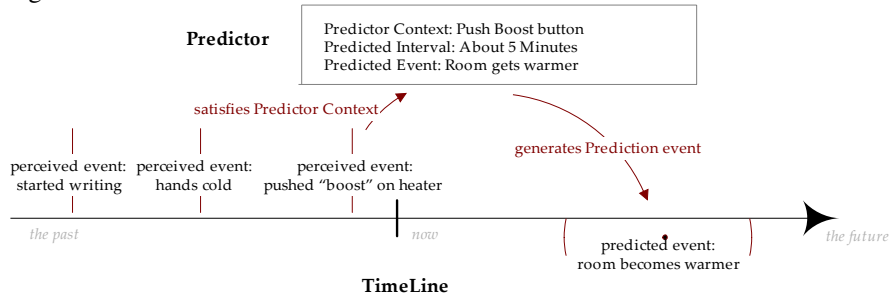
**Fig. 8.** TimeLine for a non-virtual agent (me).

So here we have a timeline that shows some of the salient events of my past few minutes. Above the line are Perceived Events, which are the things I actually observed to occur. Importantly, we assume that the perceptual input has been filtered so that only the most salient events are placed on the TimeLine.

Below the TimeLine are Predicted Events, which are the things I predicted to occur. I pushed the 'boost' button on the heater about three minutes ago, expecting that after an interval of about ten minutes, the room would get warmer. So now, three minutes later, I expect that in five minutes or so, I'll no longer be shivering as I type this. This is an example of *apparent temporal causality*: I have no idea how the system that heats this room works, but I'm pretty sure that since I pushed the 'boost' button on the thermostat that controls the heater, I'm going to feel warmer soon.

This is the simplest and perhaps most typical case of an expectation: some starting conditions are perceived that cause an expectation of an event that is predicted to occur at some time in the future. It's simple to build a representation for relationships like this. We'll call the starting conditions our *Predictor Context* – the context of the world that allows us to make the prediction. The event we expect to occur is the *Predicted Event*. The interval between the Predicted Event and the

time the prediction is made (when the Predictor Context became active) is the *Predictor Interval*. Rather than having a specific time at which the future event is likely to occur, like in the above example it is best to represent it as a window, as Fig. 9 illustrates.



**Fig. 9.** Five minutes ago, this Predictor started a Trial and caused the expectation of a future event (the temperature in the room will increase).

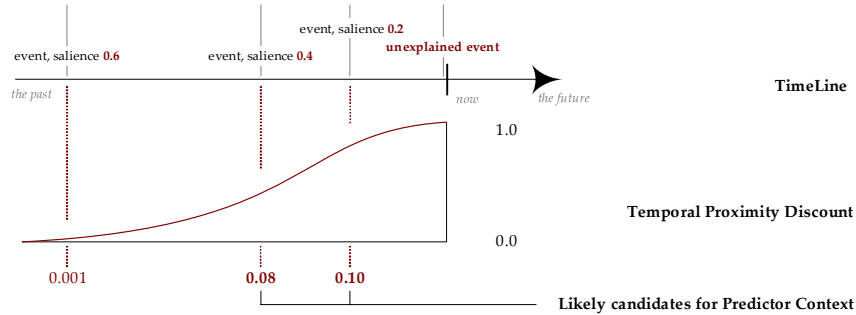
So this Predictor represents our understanding of the relationship between interacting with the thermostat, and the future temperature of the room. Imagine a Predictor as a sort of sentinel that waits around, watching the salient Perception events on the TimeLine, until its Predictor Context becomes active. When it does, it interacts with the TimeLine to “start a Trial” by forming an expectation.

The Predictor then monitors these newly-active Trial that each represent an instance of an expectation. The name Trial comes from the fact that every expectation the creature makes can possibly succeed or fail. If a Perception event equivalent to the Predicted Event occurs on the TimeLine during the window outlined by the Predicted Interval, the Trial is marked a success. Otherwise, the Trial is marked a failure. (There is also the possibility that a lack of success can be explained by some other mechanism. See [1], [7].)

### Predictors come from surprises

Where do these Predictors come from? In some entertainment applications, it would be sufficient to pre-program an agent with a set of Predictors that describe how its world is going to work.

However, as hinted at above, surprises can motivate an agent to form new Predictors. A surprise is *any salient Perceptual Event that can't be explained by a Predictor*. Understanding why surprising events occur may help the agent predict similar salient events in future. So, when a surprising Perceptual Event occurs, the creature creates a new Predictor to “explain away” the event [29]. The Predictor thus formed is a sort of “hypothesis” that the agent will be able to test in future. One way to form this hypothesis is to look back over the TimeLine for another salient event that occurred recently, and choose it as the Predictor Context, as shown in Fig. 10.



**Fig. 10.** One technique for selecting a Predictor Context. This process creates a “hypothesis” Predictor to try to explain a surprising event.

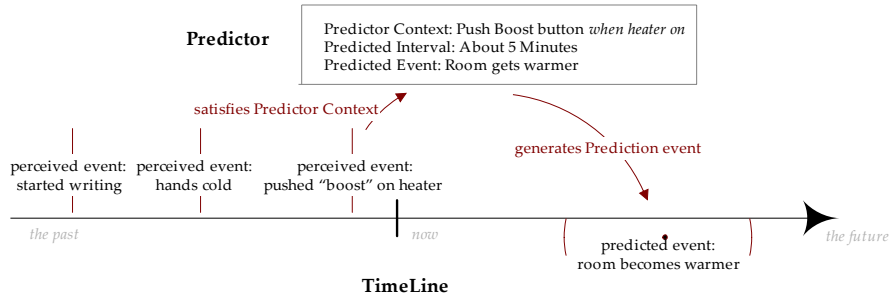
The ideal Predictor Context is one that is a *salient* and *reliable indicator*. It must be salient, so that it is likely to be noticed by the agent in future. And it must be a reliable indicator, meaning that when it *is* perceived, it follows that the Predicted Event is likely to occur; and, importantly, when it *doesn't*, the Predicted Event is *not* likely to occur.

Whatever technique we choose for creating Predictor hypotheses has to be probabilistic in nature. The mechanism is going to make many mistakes, and many Predictors formed in this way will need to be culled.

### Predictors learn from expectation violations

Sometimes, the first hypothesis doesn't tell the whole story. Perhaps I've learned that when I push the “boost” button on the thermostat, the house warms up in about ten minutes. But one day, I push the button, and half an hour later, I'm still cold. I experience an *expectation violation*, and my belief in the reliability of the Predictor decreases. Maybe my hypothesis – that pushing the “boost” button on the thermostat will warm the house – isn't always true!

So I start recording (remembering) the more salient components of the world's context that I perceive at the start of the “trials.” Some days, my attempt to warm the room is successful, and other days, it's not. After a while, I come to realize that there *is* a reliable indicator of whether or not this technique will heat the room: the switch beside the thermostat needs to be in the ‘up’ position. And so I refine my hypothesis to say that if the switch beside the thermostat is flipped ‘up,’ and I push the boost button, the room will be warm in about five minutes.



**Fig. 11.** Predictor with an updated Predictor Context.

And this, unfortunately, is a true story. (I thought it was a light switch.) What it serves to illustrate, apart from my occasional acute lack of adaptive ability, is that we can *refine* Predictors to be more useful by looking for other salient and reliable indicators that should be included in the Predictor Context.

### The Cognitive Economy

Creatures built in “wetware” (like ourselves) are limited by the number of neurons that can fit in our skull cavity. Virtual creatures are similarly limited by resources like memory and processing power. The result is a sort of cognitive economy, where for every source – the Predictor production mechanism, for example – there must also be a sink. It is not obvious when to “cull” Predictors, because sometimes negative knowledge can be extremely useful, as Minsky describes in *Society of Mind* [24]. Nevertheless, a mechanism must exist for culling Predictors that are consistently unreliable. This may result from a change in the world, or by a mistaken initial hypothesis.

Mechanisms already exist in the architecture for allowing the agent to be judicious when creating structures like Predictors. For instance, only salient perceptual information is processed on the TimeLine. (The salience heuristics, facilitated by the hierarchical structure of the Percept Tree, account for extreme sensory input, sudden and pronounced changes in sensory input, and sensory input previously associated with high-valued actions.) Predictor refinement is also controlled by focusing on Predictors with unusually dynamic reliabilities. Interestingly, there is evidence that unreliable reinforcement leads to exploration in real animals as well [14].

### To summarize, the creature wants to understand how its world works

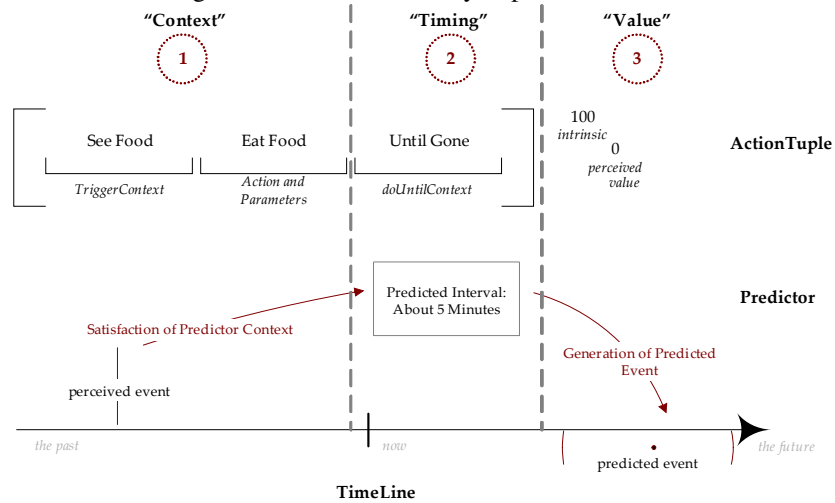
The thermostat example serves to illustrate the principle that drives predictive learning: that an agent should seek to understand enough about the world to predict things that it would find useful. The mechanism described here is guided by salient events to form hypotheses that are represented as of Predictors, which are then verified and refined.

This particular model of prediction, and particularly the inclusion of windowed timing information within the model, was inspired by a recently-proposed ethological model for classical and operant conditioning in real animals ([14], [7]).

**But what happened to Action?**

But you may have noticed that in our discussion of prediction, we didn't talk at all about how the Predictor representation affects how the agent makes action-selection decisions. The Predictors help predict events that are going to occur in the future, but how does that relate to what actions the agent chooses to take?

There's an intriguing isomorphism between the ActionTuple and the Predictor. When we talked about action selection and the ActionTuple representation, it was shown to be useful to break down our representation for action into the *TriggerContext*, which contains the external conditions that caused an action to be relevant; the *Action* and its *Parameters* that describe the self-action required by the ActionTuple; the *DoUntil*, which describes the length of time the action would take; and finally the *Value*, both *Intrinsic* and *Perceived*, of performing that action in the given context. The Predictor is similarly broken down into a *Predictor Context*, which contains the conditions that cause the Predictor to become active; the *Predictor Interval*, which contains the expected length of time between the predictor context and an upcoming event; and the *Predicted Event*, which is really the "value" of having the Predictor – its ability to predict a future event!



**Fig. 12.** Isomorphism between ActionTuple and Predictor representations: both encapsulate Context, Timing and a sense of "Value."

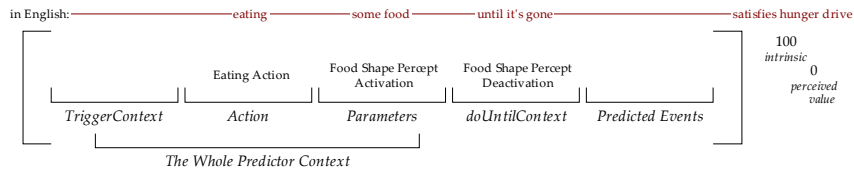
In other words, both contain (1) some sort of triggering mechanism based on context, (2) a timing mechanism, and (3) some sense of future value. By now it is obvious that there must be some way to take advantage of this relationship to in-

fluence action selection, and indeed there is. The final clue for one useful action-selection scheme comes from the fact that the Predictor Context occasionally, *but not always*, include an element of self-action. What if the action *itself*, and its parameters (like the object on which you’re performing the action) became *part of the context*?

If we replace the Value component of all ActionTuples with a Predicted Event, then the ActionTuple itself *becomes* a Predictor. The Predictor Context is now composed of the *TriggerContext*, the *Action* and its *Parameters*! Both literally and figuratively speaking, the “Value” of this “Predictive ActionTuple” is *its ability to predict the Predicted Event!*

### Learning with Predictive ActionTuples

To show that this system is sufficient to reproduce a back-propagation learning technique, let’s consider how the dog training example works with Predictive ActionTuples. The dog still has one consummatory Predictive ActionTuple, augmented as described above, and shown below in Fig. 13.



**Fig. 13.** A Predictive ActionTuple. This is the dog’s consummatory Predictive ActionTuple. Note that we start being more explicit here about the fact that all the components actually refer to Percepts. The “Eating Action” even maps on to an EatActionPercept. The Predicted Events slot will be discussed in a moment.

At some point, the agent unexpectedly perceives food – something that lets the above Predictive ActionTuple be activated. But why did food appear? Perhaps something happened a moment ago that, in future, could serve as an indicator that food is again on the way.

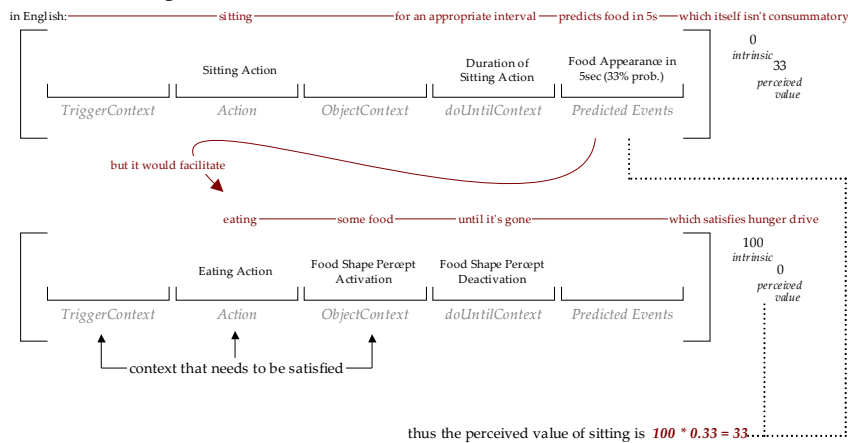


**Fig. 14.** Timeline that shows the agent sat down a few moments prior to the unexplained appearance of food.

According to the TimeLine shown in Fig. 14, the most salient thing that happened recently was self-action – the agent just sat down. So perhaps sitting in the future will reliably be followed by the appearance of food. This can be represented as a Predictive ActionTuple.

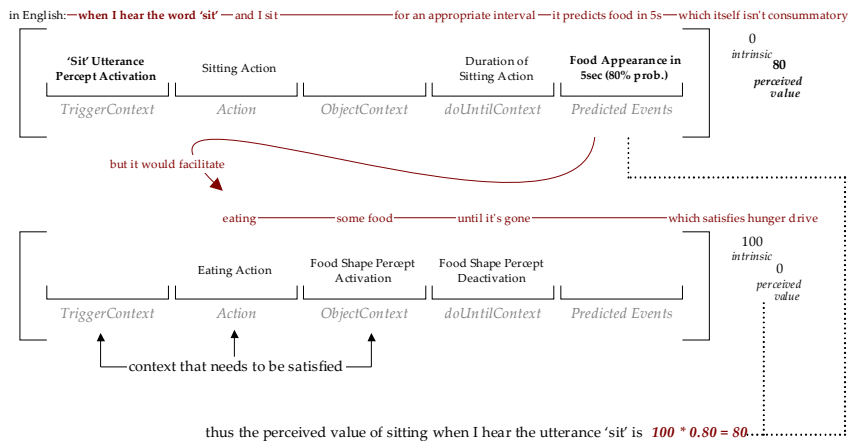
It is important to note that the perception of food is not inherently useful. However, the agent has another ActionTuple – the “eat food when it’s present” ActionTuple depicted above – that is intrinsically valuable when the agent is hungry. Thus, in the new system, our new Predictive ActionTuple obtains value not

because it has “Perceived Value” back-propagated to it, but rather because of its ability to predict the appearance of the stimulus that would allow for the activation of an intrinsically-valuable Predictive ActionTuple. In other words, it allows us to “complete the context” necessary to activate the intrinsically valuable Predictive ActionTuple. (A nice side effect of this is that if food is already available, the perceived value of sitting is reduced, because, although that may cause food to appear, we already have the food necessary to complete the context of the “eat food” Predictive ActionTuple.)



**Fig. 15.** The two Predictive ActionTuples working together to create perceived value. The Predictive power of the mechanism described above has been harnessed for action-selection purposes.

Just like in the thermostat example, this new causality relationship (sitting leads to food) isn't as well-defined as possible. The agent will soon find that the probability of food appearing after sitting down isn't particularly high. However, there will be some reliable thing about the context at the start of each Trial that predicts, with great reliability, the Trial's success or failure. (More detail about techniques for tracking salient stimuli during Trials is found in [7].) If there is an utterance that sounds like “sit,” and the agent sits down, then in a few seconds, food will appear with a certain probability. So the context of the Predictor is refined to reflect this realization.



**Fig. 16.** The Predictive Action Tuple now only makes predictions (starts Trials) when the agent hears the word sit and sits down. Thus, both external context and self-action are required to trigger a prediction.

The “appetitive” version of a Predictive Action Tuple is valuable because it predicts some future event that will facilitate the onset of an intrinsically valuable (or “consummatory”) Predictive Action Tuple!

Now, the agent knows exactly why it should sit when it hears “sit” – because it expects that food is going to show up if it does! So when it hears “sit” and does sit down, it can drop a Predicted Event on the TimeLine and anticipate the upcoming food. If it does come, the agent is happy – but not surprised. If food doesn’t come, there’s an expectation violation, and the agent is able to wonder why.

### The Goatzilla Domain and new Forms of Behavior

The Predictive Action Tuple mechanism was implemented and tested for a virtual creature reminiscent of many found in entertainment applications. Allow me to introduce Goatzilla, a behemoth of an autonomous virtual creature that inhabits the Scottish highlands.



**Fig. 17.** Goatzillas in the mist.

Goatzilla's most important drives are hunger and the dominance of other beasts, and his food source is the shed in which the local shepherd keeps his sheep. If Goatzilla kicks the shed, the frightened sheep scatter, and he can embark on a feeding frenzy. Among his other drives is curiosity, which rises slowly over time, and can be reduced by interacting with unusual objects, performing less-familiar actions, and experimenting with Predictor relationships that have a high degree of entropy (the ones that would sometimes, but not always, result in success).

The fundamental action selection decision an agent like Goatzilla has to make at every moment is whether to explore, exploit or react. In the mechanism shown here, the agent first decides whether it will explore or exploit. Goatzilla does so by performing "drive selection," wherein one of the high-level Drives is probabilistically chosen, using a distribution weighted by the Drive Multipliers ((1) in Fig. 18). If the mechanism chooses *any drive other than curiosity*, it selects an action that, in addition to being generally good for satisfying all drives, should be particularly effective at reducing the selected Drive. Thus, the creature *exploits* its existing knowledge as best they could.

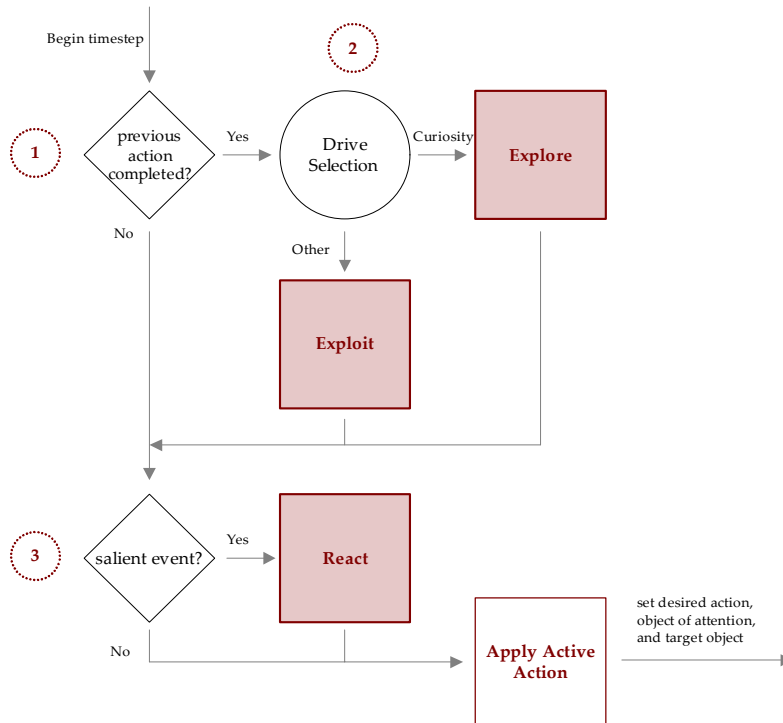


Fig. 18. Goatzilla’s Action Selection Mechanism. (Simplified; see [7] for more detail.)

However, if the curiosity Drive is chosen, the agent chooses to *explore* rather than *exploit* ((2) in Fig. 18). Instead of having exploration strategies scattered throughout the system, it was useful to assemble many of them into this central mechanism that could be called upon to choose an appropriate exploration strategy. For example, the Exploration mechanism sometimes would cause the creature to activate a Predictive ActionTuple that had been recently unreliable, so that the creature could initiate a new Trial for that ActionTuple’s Predictor, and thus further test the validity of that hypothesis. Or, it could generate a new ActionTuple by taking an existing successful ActionTuple (kicking the shed makes food appear) and modifying it to make another ActionTuple hypothesis (perhaps kicking a rock makes food appear? ...or what about kicking any building made of wood?).

If an unexpected stimulus is perceived, it may be possible to interrupt the creature’s current behavior to *react* ((3) in Fig. 18). If food appears, for example, we might immediately interrupt our current behavior to approach and eat it. Or, if Goatzilla is wise in the ways of the world and perceives a shed somewhere out in the mist, he may interrupt his behavior to approach and kick it, because he knows this will let him “complete the context” and thus activate the Predictor that says sheep (read: food) will appear momentarily.

Predicted events can also be used by the action selection mechanism to produce *anticipatory* and *avoidance* behavior – something that would be impossible without an understanding of causality. For instance, if we have learned (like one of Pavlov’s dogs) that the ringing of a bell is reliably followed by the presentation of food, then upon hearing the bell we might react by *anticipating* the appearance of the food [25]. On the other hand, if we’ve learned that the ringing bell is reliably followed by an electric shock, then hearing the bell ringing would drop a prediction of the shock on the timeline. As the expected time of that shock approaches, the agent might react by attempting to *avoid* the onset of the shock. In Goatzilla’s implementation, special “approach,” “avoid” and “observe” ActionTuples allowed him to accomplish these reactive tasks.

Here is an interesting example of a behavior facilitated by the representation for causality. In the absence of a relevant stimulus, the creature can come up with a strategy for satisfying its drives. If Goatzilla’s hunger drive is high and he is asked to Exploit, he can ascertain that, ideally, he would like to be eating. But in order to activate the “Eat Sheep” ActionTuple, he is missing a prerequisite: the Sheep. He can reason that, with high probability, kicking the shed will lead to the appearance of sheep in a short time. Thus, he could approach and kick the shed, dropping onto the TimeLine the prediction that food will appear in a few seconds. After doing the kicking, he can stand back and anticipate the appearance of the sheep. If they appear, *he is not surprised*; rather, he feasts, and reinforces his “Kicking the shed causes sheep to appear” Predictive ActionTuple hypothesis. If they do not appear, *there is an expectation violation*, he is *disappointed by the absence of an anticipated perception*, and he reduces his confidence in the hypothesis.

As seen in this example, the TimeLine representation provides all the information the action-selection mechanism needs to make these *explore*, *exploit*, *react* decisions. To evaluate the Value of a Predictive ActionTuple, the agent can use the future portion of the TimeLine to generate a counterfactual – imagining what the world would be like if a particular action was performed by adding the Predictive Events from that ActionTuple to the TimeLine. (See [15] for an intriguing discussion of counterfactuals). The TimeLine is also useful for determining if recent Perceived or Predicted Events might offer the agent a chance to interrupt its current behavior and *react*.

### **Socially-Oriented Prediction**

The inclusion of a virtual hockey player as an example of an autonomous agent at the start of this chapter was motivated not only by the author’s Canadian citizenship, but also because it illustrates an agent acting in a social setting. Agents with social understanding can also benefit from a mechanism that predicts how *other agents* (real and virtual) will react to a context.

Elsewhere in this book are thorough examples of socially-oriented agents. In *Society of Mind*, Minsky uses Scripts as a representation for a social activity (such as a birthday party, a conversation, or a play in a hockey game) in which more than one agent participates [24]. These scripts imply social protocols which de-

scribe how each agent expects the others to behave in a particular situation, given each agent’s role in the social group.

In the hockey game, for instance, the agents are members of the social group known as a hockey team. At a particular moment, the six players on the ice may each be acting autonomously to implement a “behind-the-net” offensive strategy. Typically used by teams with strong puck control, the execution of this strategy requires meticulous teamwork – in other words, the ability to predict the reactions of your teammates to a dynamic and relatively uncertain situation.

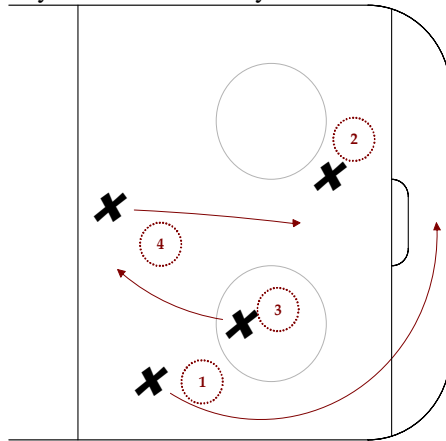


Fig. 19. The “Behind-The-Net” Offensive strategy, as described below.

In the strategy, the puck-carrier’s goal ((1) in Fig. 19) is the maintain possession of the puck and bring it behind the opponent’s net. As he does so, he relies on his two offensive teammates ((2) and (3)) to position themselves in front of the net to set up possible “one-time shots” – quick shot opportunities that are likely to fly past the opposing team’s goalie. As the puck-carrier crosses behind the net, this triggers the defensemen (4), who are waiting at the blue line, to rush towards the net in anticipation of a pass that might create an exceptionally dangerous shot opportunity. If necessary, at that moment the offensive teammates (e.g. (3), as shown) can move away from the net to prevent the opposing team from breaking away with the puck. (I am indebted to [12] for this description.)

This strategy could be represented as a branching script, in which each of the participants – the six teammates, and their six opponents – have defined roles. So long as each agent can trust that each of the other agents on his team knows a similar “script,” he can predict how the others will react to the current context. The puck-carrier, for example, can predict that if he can cross behind the net, he can expect the defensemen to rush towards the net to produce a one-time shot opportunity. And such an opportunity is likely to lead to a goal.

All of this can be represented using Predictive ActionTuples and the TimeLine mechanism. The additional requirement, which is beyond the scope of this chapter (and the current implementation), is a set of Predictors that base their predictions on the state of currently active social scripts.

It is worth noting that in current entertainment applications, such as sports simulators, a more computationally-efficient approach for such social behavior would involve a “meta-agent” that controls the entire hockey team, issuing orders to individual agents rather than considering the players as autonomous agents. However, such an approach would have to work hard to produce the illusion of life provided by limited agent-based perception. Certainly it could achieve this effect by determining, at the meta-agent level, what each individual agent is able to perceive. If the arena is viewed from 10 meters off the ground, such details are likely to go un-noticed. But during dramatic close-ups and replays, the illusion of life would be greatly augmented by authentic touches like things like the look of surprise on a player’s face as an opponent he failed to notice steals the puck out from under him.

### **Devils in the Details**

Although the technique has proved highly effective for creating an agent that can learn about causality while simultaneously behaving in a lifelike way, there are a couple of subtle nuances that should be mentioned.

Evaluating the Perceived Value of a Predictive ActionTuple is a recursive process. This makes it possible to for an agent to generate predictions based on complex causality chains, but it also introduces a potential computational bottleneck. Perhaps a superior model of attention would allow for a dynamic limit on the recursion depth.

We haven’t discussed how to organize ActionTuples, either of the Predictive or regular variety. As an agent gets more complex and able to adapt to and interact with more contexts, this becomes increasingly important. Depending on the application, context-action pairs might be organized into groups based on some higher-level context. Perhaps this would be in terms of higher-level goals (like winning a hockey game, versus discussing the match afterwards). Current research involves techniques for organizing actions in terms of social function (things I do with my family, things I do with my hockey team, and so on).

### **Things That Worked**

**Causality and Action Selection are integrated.** To take advantage of causality information, the contexts an agent uses to trigger its actions can be equivalent to the contexts used to trigger predictions.

**A desire to understand the world drives learning.** When an event occurs that the agent doesn’t predict, it registers *surprise* and invents an explanation for why the surprising event occurred. When an explanation (in the form of a Predictor) turns out to be erroneous, an *expectation violation* occurs and the creature can either refine the explanation or invent a new one. In the absence of unusual stimuli, the agent’s *curiosity drive* motivates it to explore. It bears repeating that all three of these fundamental motivations for learning emerge from a desire to *understand the world*. Certainly, the agents described here are highly motivated to

satisfy their drives. But, instead of perpetually attempting to maximize rate of return, these agents instead seek to understand *enough* about the world to satisfy their drives effectively and predict the onset of salient events. They are *then* motivated by curiosity to discover new things, some of which may lead to new techniques for maximizing rate of return. One observer called this the “curious slacker” approach. The results suggest it creates creatures that are better able to sustain the illusion of life, and avoid some local maxima in the process.

**There is a Cognitive Economy.** For every source there must be a sink. For every mechanism that deposits topology, there must be a mechanism that performs withdrawals. The architect of a cognitive architecture must consider the performance of the system on various time scales – eight seconds, eight minutes, eight hours, eight days – as well as in the theoretical limit. What will happen, for example, to knowledge that is rendered useless by a change in the environment?

**Nothing is deterministic.** It is almost always the case that when some decision in an agent is deterministic and involves selecting the “best option,” the agent is afforded an opportunity to get stuck in a mindless loop. It invariably will. Some mechanisms, such as the hypothesis-generating mechanism for Predictors, obviously require a random element. Others, like the one underlying the *exploit* operation that is meant to select the “best” option, should also employ some degree of randomness. Choosing an optimal and intuitive distribution for a selection process is another story. The theory of probabilities, noted Laplace, “is nothing more than good sense confirmed by calculation.”

**The agent’s representation of the world is (slightly) less simplified.** Many credit assignment and machine learning algorithms make substantial assumptions about how the world is represented. If we are overzealous in our attempts to simplify our mental representations of the world, we risk introducing what McCallum calls *aliasing* – the inability for adaptive representations in the system to learn the right things, because the perceptual representations can’t distinguish between the things they need to learn about [23].

## Conclusion: What we get from prediction

A representation for prediction and expectation can be added to a system designed to support creatures that maintain the illusion of life. The mechanisms are capable of adaptation, so that a creature built using the system can learn about causality “on the job” and even react to changes in their world. In addition to the new motivations for learning that are facilitated by surprises and expectation violations, this system also extends an agent’s capacity to emote, by providing a plausible source for emotional effects like eager anticipation or dread, satisfaction or relief, and many others. I would not be surprised to see prediction become increasing pervasive in agents built for entertainment applications.

## Acknowledgements

I am indebted to my former advisor Bruce Blumberg and the Synthetic Characters group at the MIT Media Lab, as well as Whitman Richards and Randy Gallistel. I also thank Scott Eaton, Gary and Phil McDarby, Daragh McDonnell and the rest of the MindGames group at MediaLabEurope, who helped bring Goatzilla to life. Twice.

## References

- [1] Allen, J. F. (1991). Planning as Temporal Reasoning. The Second International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, MA, Morgan Kaufmann.
- [2] Blumberg, B. M. (1996). Old Tricks, New Dogs: Ethology and Interactive Creatures. Media Lab. Cambridge, MIT.
- [3] B. Blumberg, M. Downie, Y. Ivanov, M. Berlin, M. P. Johnson, B. Tomlinson. "Integrated Learning for Interactive Synthetic Characters." In: Proceedings of SIGGRAPH 2002, ACM Press.
- [4] Brooks, R. A. and Lynn Andrea Stein "Building Brains for Bodies", MIT AI Lab Memo 1439, August 1993.
- [5] Brooks, R. A. (1991). "Intelligence Without Representation." Artificial Intelligence Journal **47**: 139-159.
- [6] Burke, R. C., D. A. Isla, et al. (2001). Creature Smarts: The Art and Architecture of a Virtual Brain. Game Developers Conference 2001, San Jose, CA.
- [7] Burke, R. C. (2001). It's About Time: Temporal Representations for Synthetic Characters. The Media Lab. Boston, MIT.
- [8] Cole, R. P., R. C. Barnet, et al. (1995). "Temporal encoding in trace conditioning." Animal Learning and Behavior **23**(2): 144-153.
- [9] Damasio, A. (1995). Descartes's Error. Harvard University Press.
- [10] deKleer, J. and J. Brown (1986). "Theories of Causal Ordering." Artificial Intelligence Journal **29**(1): 33-62.
- [11] Downie, M. (2001). Behavior, Animation and Music: The Music and Movement of Synthetic Characters. The Media Lab. Boston, MIT.
- [12] EA Sports (2003). NHL 2003 Coaching Strategies Videos. <http://www.ea.com/>
- [13] Ekman, P. (1982). Emotion in the Human Face. Cambridge, UK, Cambridge University Press.
- [14] Gallistel, C. R. and J. Gibbon (2000). "Time, Rate and Conditioning." Psychological Review **107**: 289-344.
- [15] Hofstadter, Douglas R (1980). Godel, Escher, Bach: an Eternal Golden Braid.
- [16] Isla, D. A. (2001). The Virtual Hippocampus: Spatial Common Sense for Synthetic Characters. MIT Department of Electrical Engineering and Computer Science. Cambridge, MIT.
- [17] Isla, D. A., R. C. Burke, et al. (2001). A Layered Brain Architecture for Synthetic Characters. IJCAI, Seattle.

- [18] Ivanov, Y., B. M. Blumberg, et al. (2000). EM For Perceptual Coding and Reinforcement learning Tasks. 8th International Symposium on Intelligent Robotic Systems, Reading, UK.
- [19] Johnson, M. P. (2002). Quixote: Quaternion-Based Techniques for Expressive Interactive Character Animation. Media Lab. Cambridge, MIT.
- [20] Kaebbling, L. P., L. M. Littman, et al. (1996). "Reinforcement learning: a survey." Journal of Artificial Intelligence Research **4**: 237-285.
- [21] Kline, C. (1999). Observation-based Expectation Generation and Response for Behavior-based Artificial Creatures. Media Lab. Cambridge, MIT.
- [22] Maes, P. (1989). The Dynamics of Action Selection. International Joint Conference on Artificial Intelligence, Detroit, Morgan Kaufmann.
- [23] McCallum, A. K. (1995). Reinforcement Learning with Selective Perception and Hidden State. Department of Computer Science. Rochester, New York, University of Rochester: 157.
- [24] Minsky, M. (1985). The Society of Mind. New York, Simon and Schuster.
- [25] Rescorla, R. A. and A. R. Wagner (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. Classical Conditioning II: Current research and theory. A. H. Black and W. F. Prokasy. New York, Appleton-Century-Crofts: 64-99.
- [26] Perlin, K. and A. Goldberg (1996). "Improv: A System for Scripting Interactive Actors in Virtual Worlds." Computer Graphics **29**(3).
- [27] Rose, C. F., M. Cohen, et al. (1999). "Verbs and Adverbs: Multidimensional Motion Interpolation." IEEE Computer Graphics and Applications **18**(5).
- [28] Russell, J. (1980). "A circumplex model of affect." Journal of Personality and Social Psychology **39**: 1161-1178.
- [29] Sheridan, T. (1992). Telerobotics, Automation, and Human Supervisory Control. Cambridge, MIT Press.
- [30] Spier, E. (1997). From Reactive Behaviour to Adaptive Behaviour: Motivational Models for Behavior in Animals and Robots. Oxford, Oxford University: 99.
- [31] Thorndike, E. (1911). Animal Intelligence. Darien, CT, Hafner.
- [32] Honda Humanoid Robot (2003). <http://world.honda.com/robot/technology/>